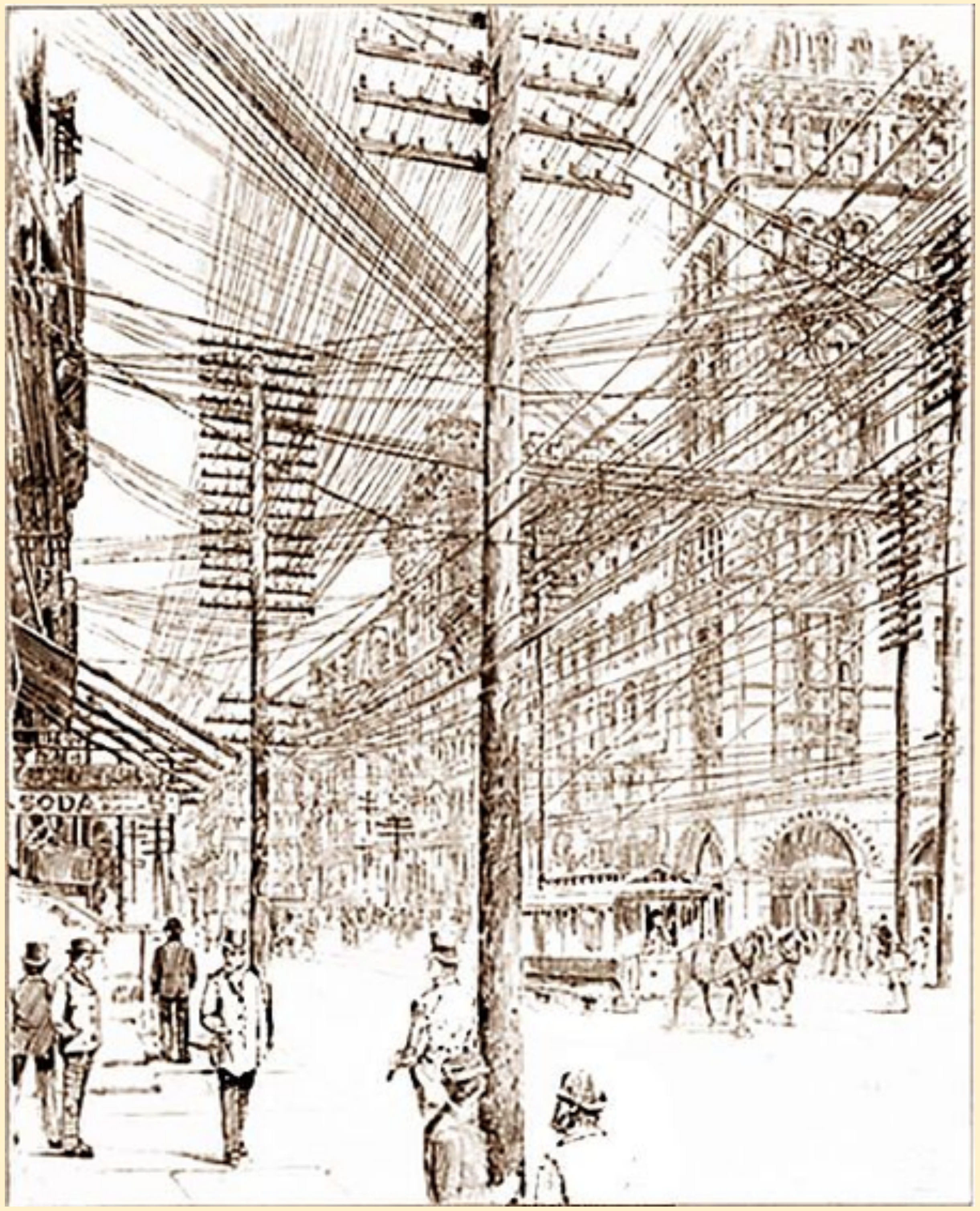
The "Last Mile" of Data Handling



Marc Mengel & Adam Lyon

Fermilab Scientific Computing Division

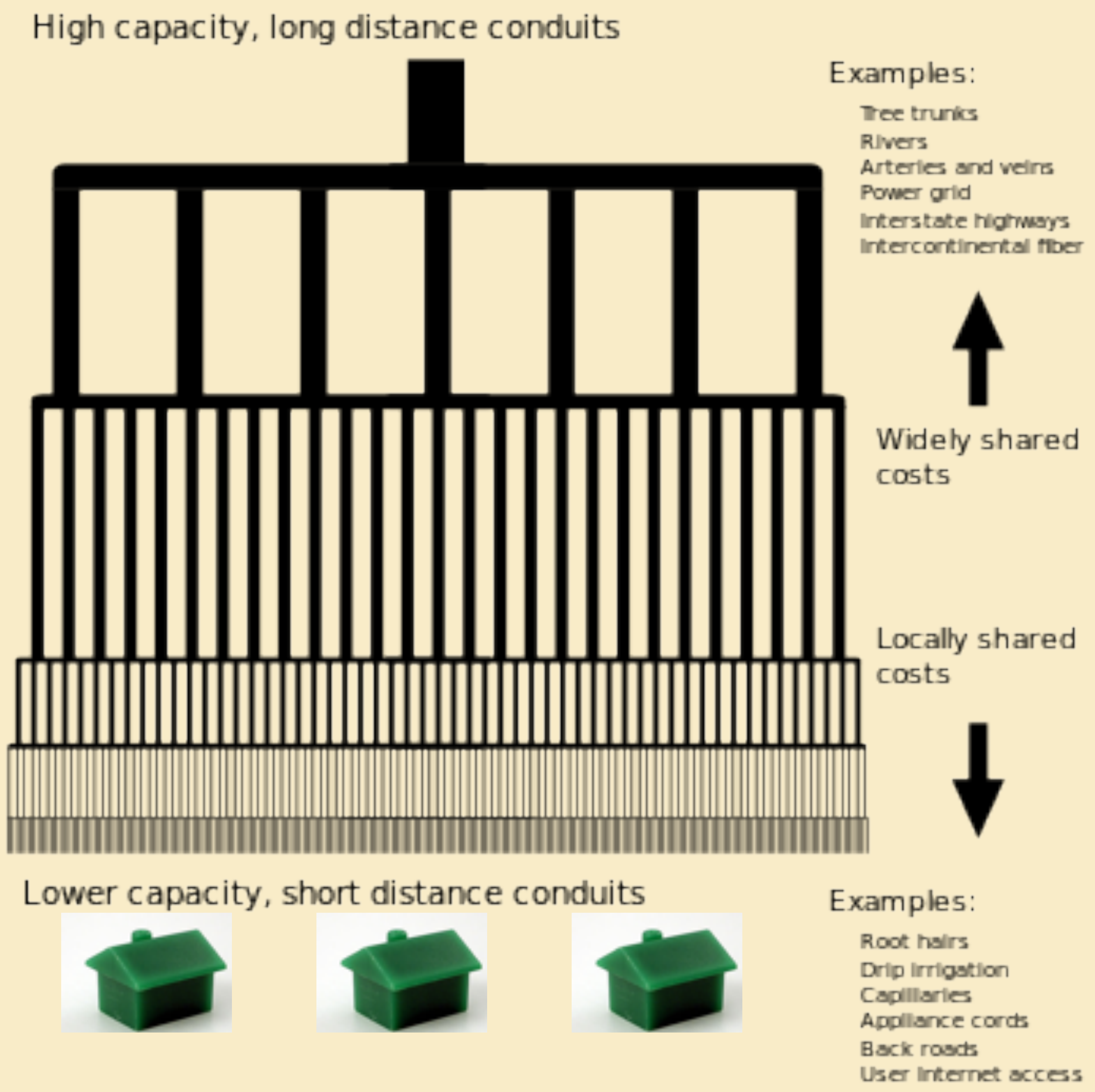
CHEP October 2013

Traditional Last Mile

The Last Mile is a common problem

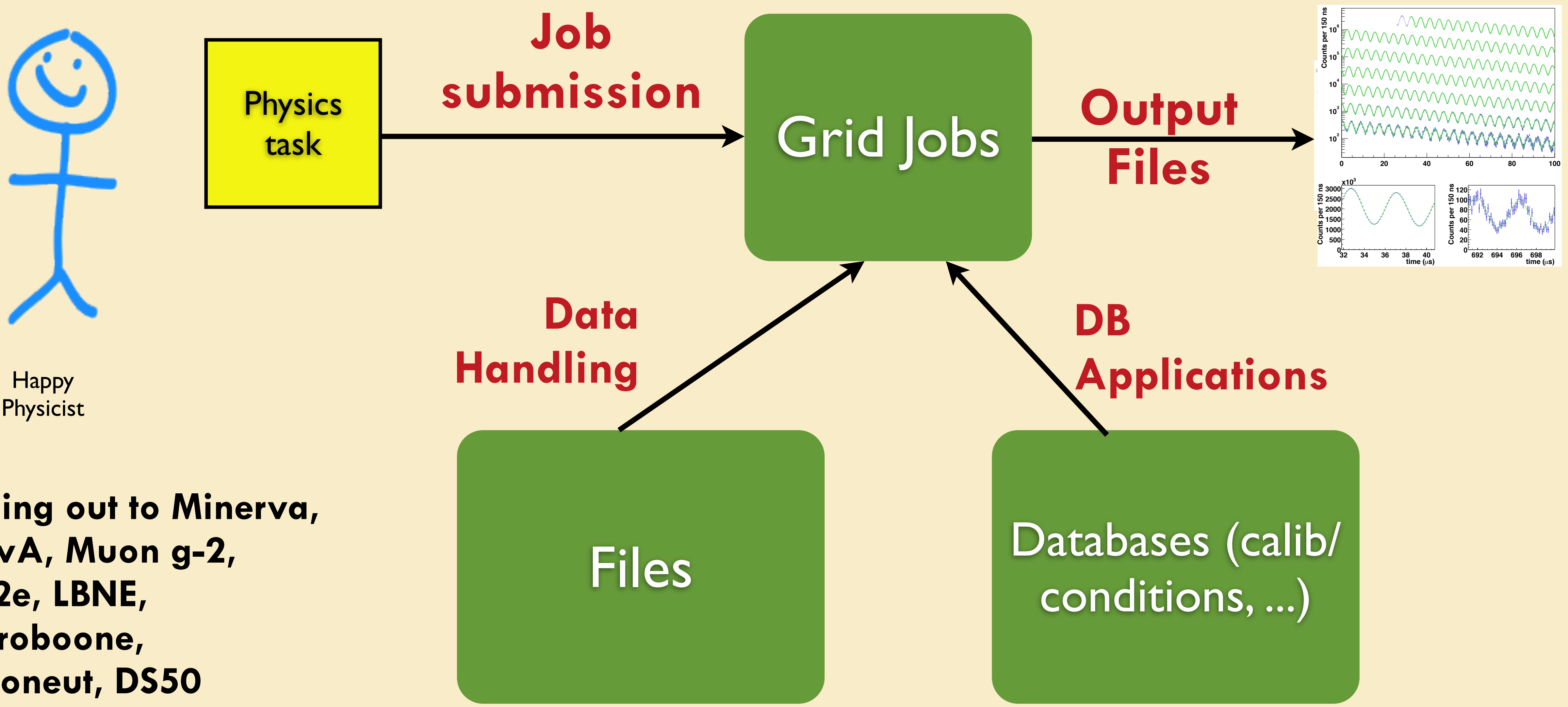
Land line telephone, power, cable TV & internet

Last mile is expensive and hard to upgrade



Our context - Moving files to jobs

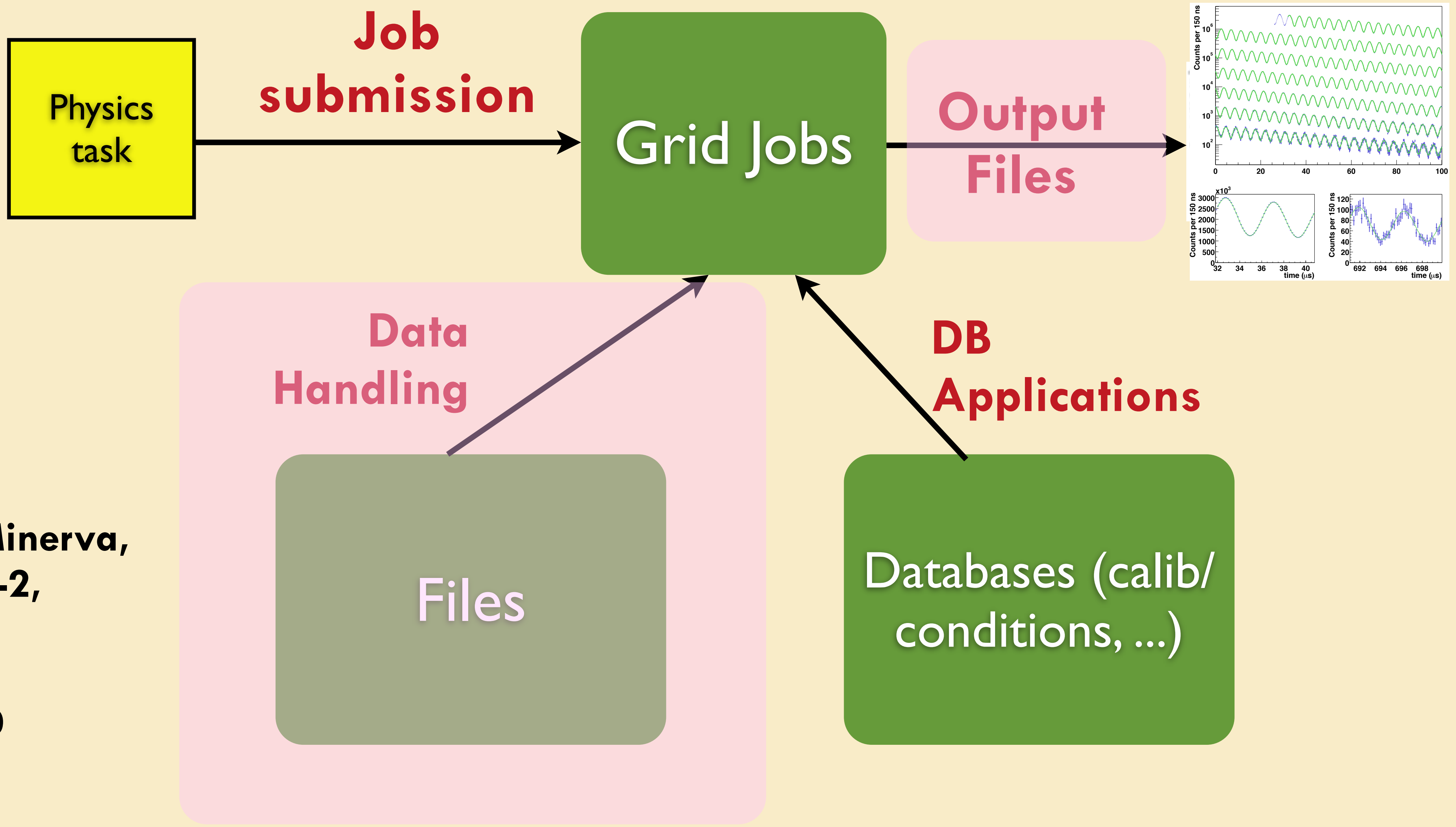
FIFE is Fermilab's overarching program of common tools and systems for scientific data processing



Rolling out to Minerva, NOvA, Muon g-2, Mu2e, LBNE, Microboone, Argoneut, DS50

FIFE (Fabrlc for Frontier Experiments)

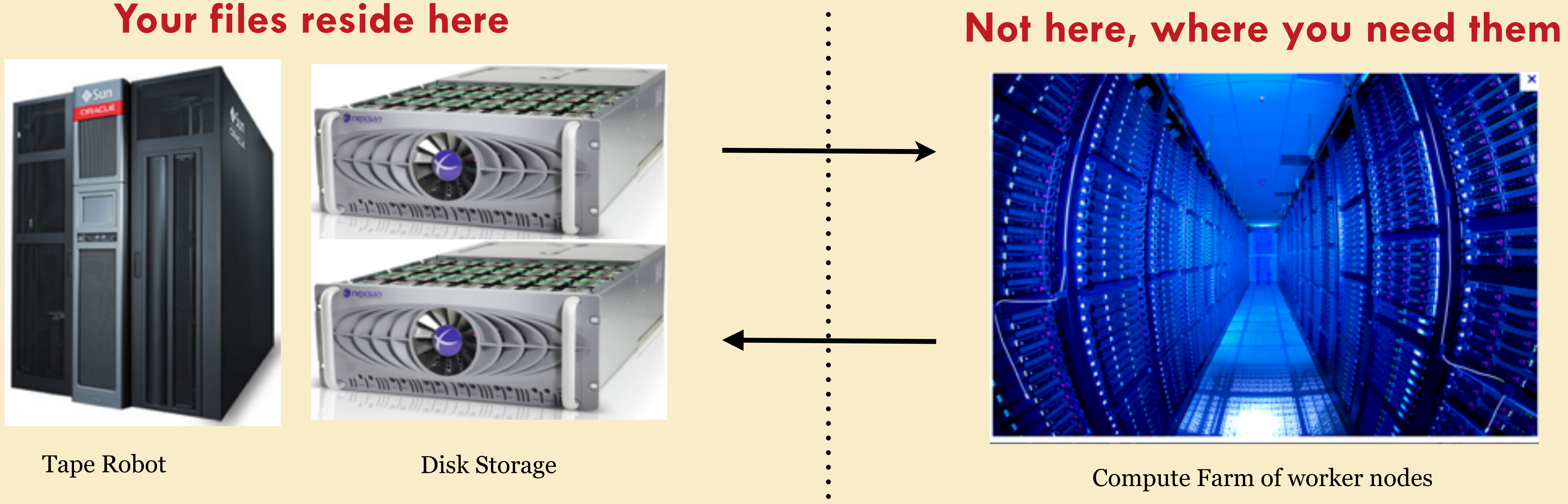
FIFE is Fermilab's overarching program of common tools and systems for scientific data processing



Rolling out to Minerva, NOvA, Muon g-2, Mu2e, LBNE, Microboone, Argoneut, DS50

Why is data movement difficult?

Your FILES don't reside where your jobs run



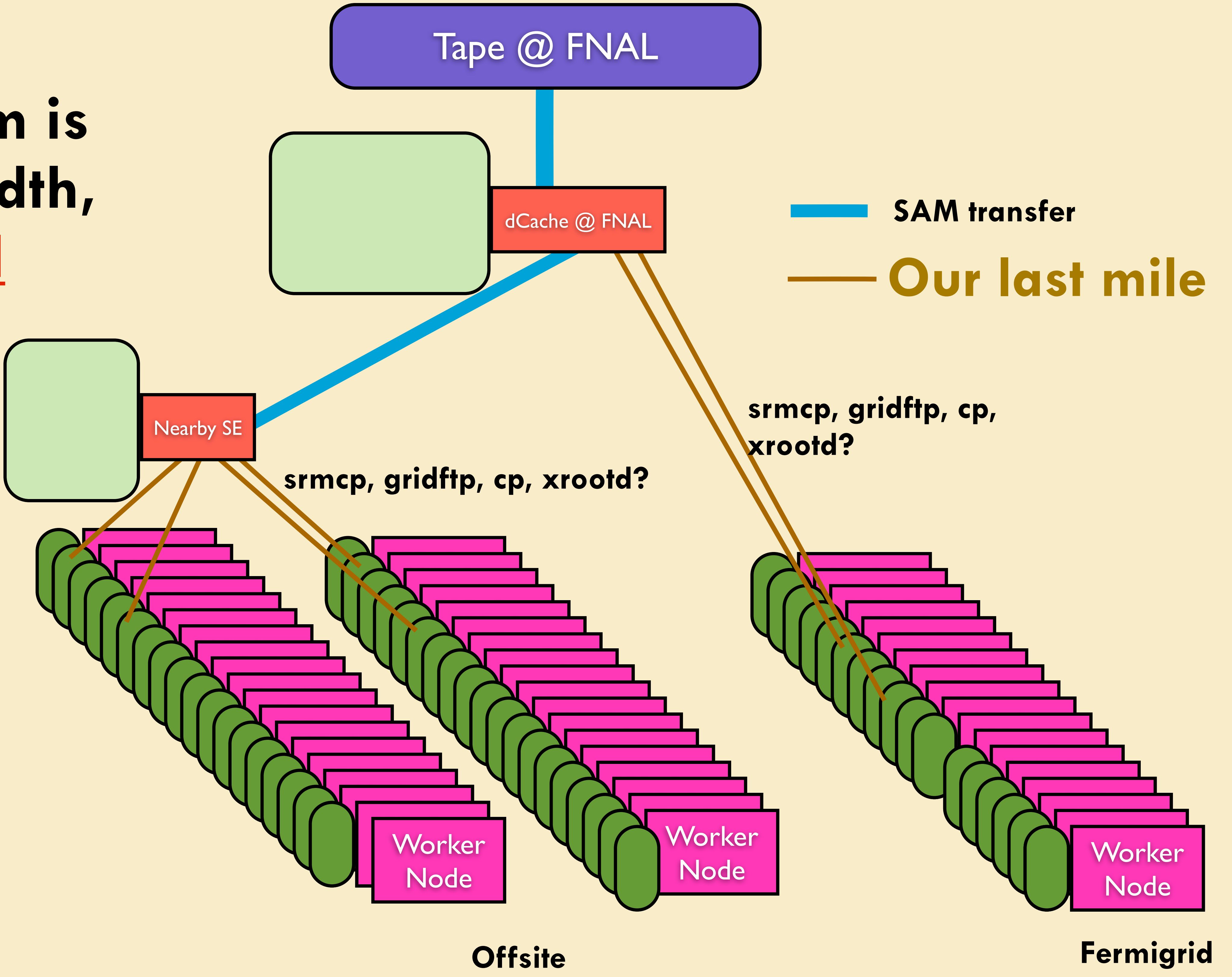
Files must be moved to where a worker node can access them with efficiency and scalability

Output files must be returned from worker node

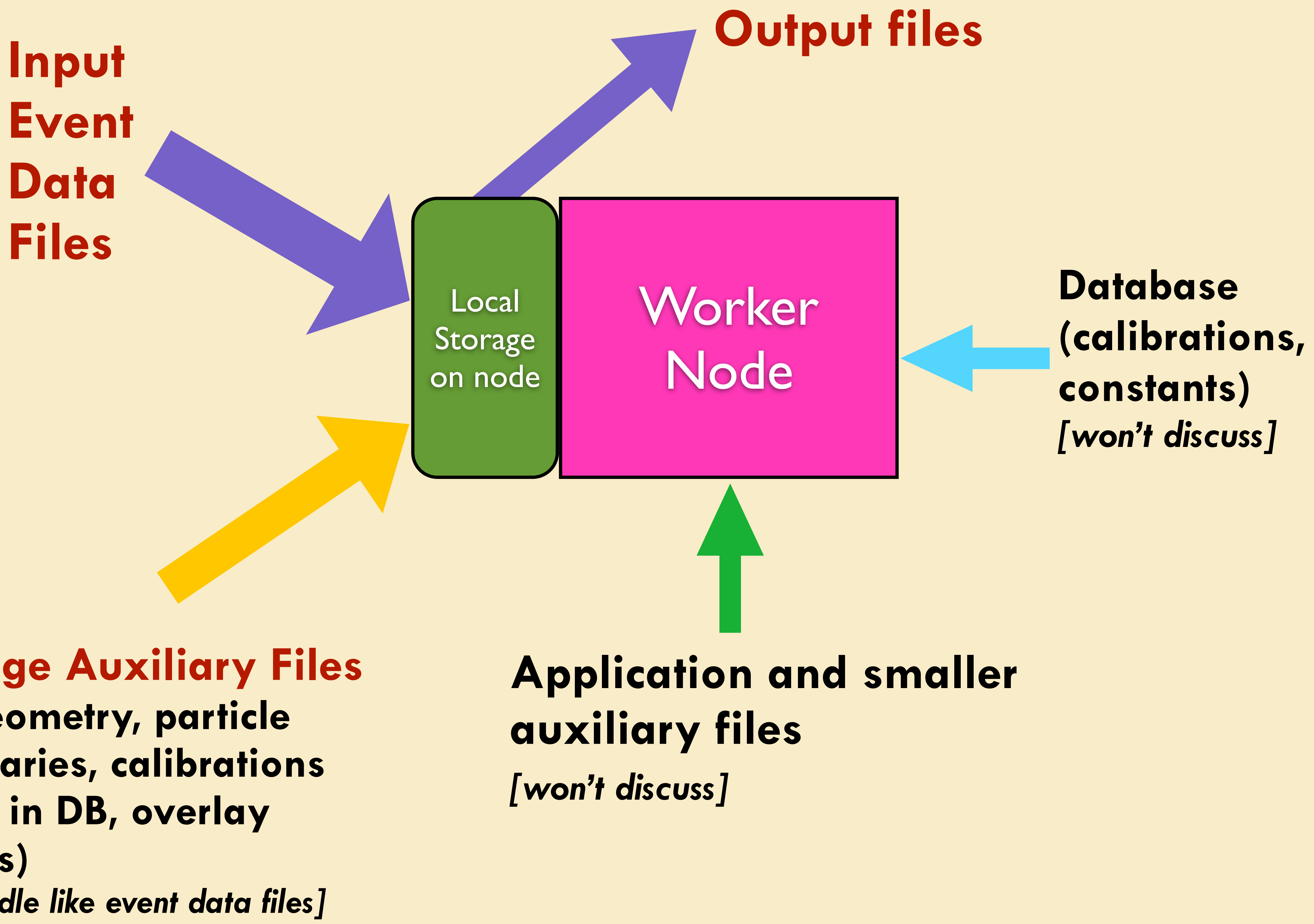
NOT a simple process, especially the efficiency and scalability part!

Our last mile problem

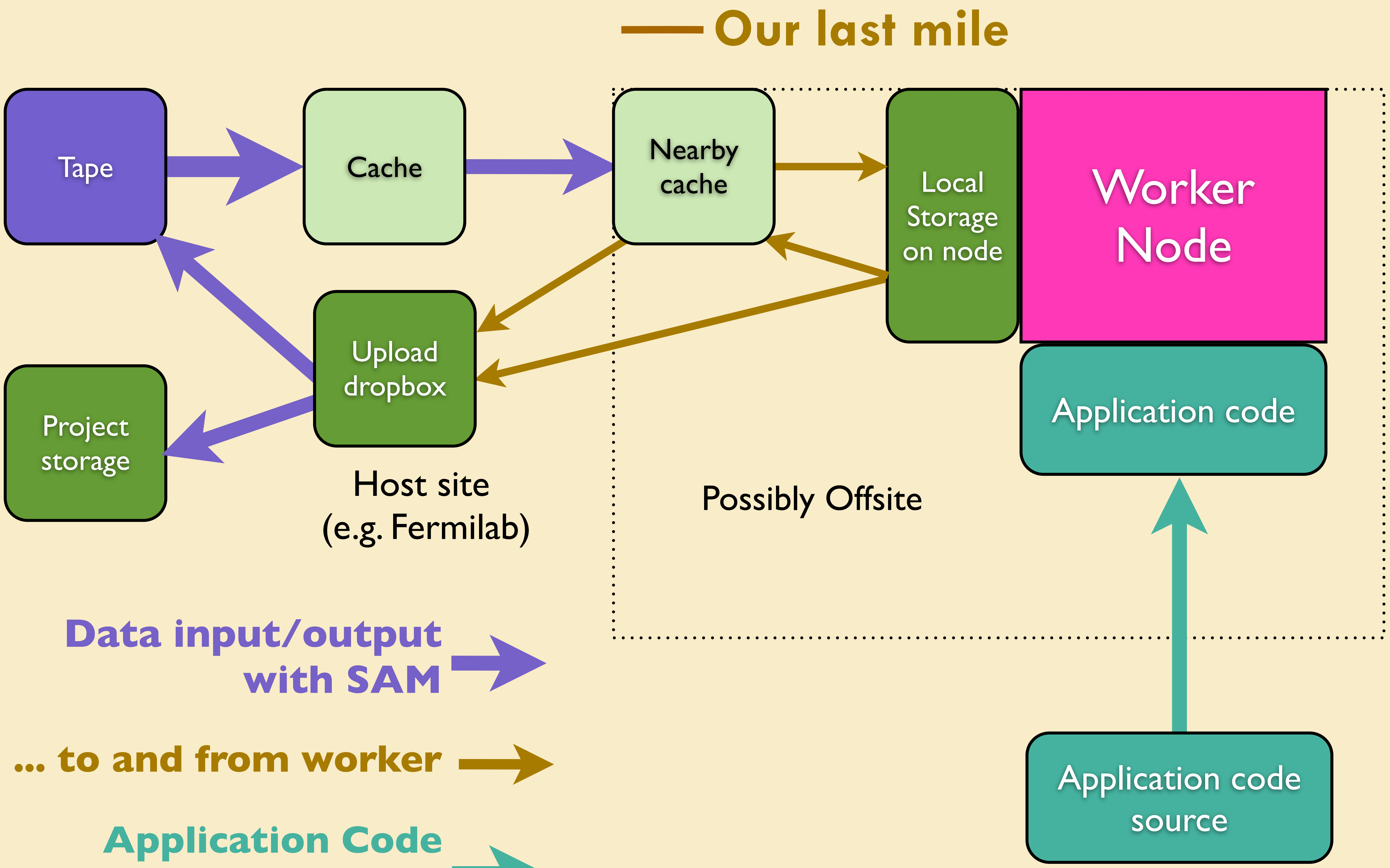
Our problem is not bandwidth, but <u>protocol</u>

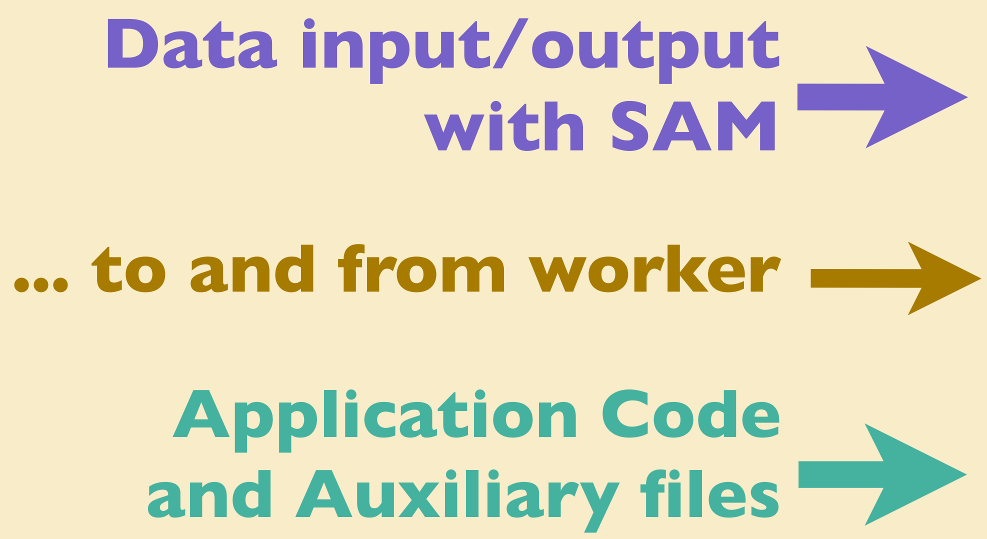


We must handle many file types

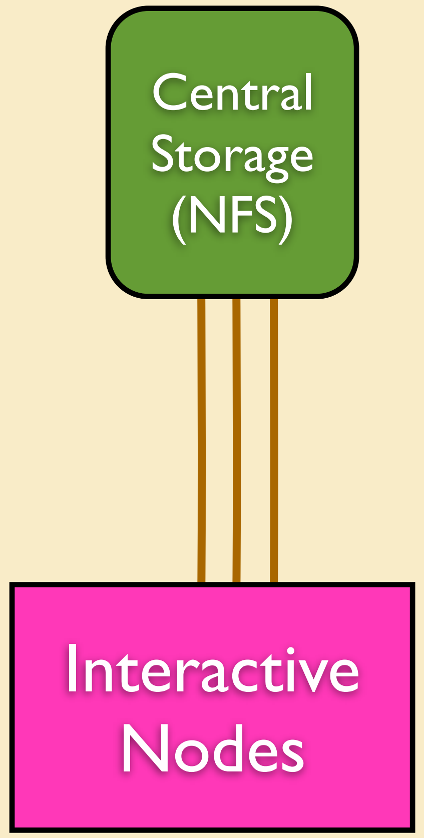

Input Event Data Files
Output files
Local Storage on node
Worker Node
Database (calibrations, constants)
[won't discuss]
Large Auxiliary Files
(Geometry, particle libraries, calibrations not in DB, overlay files)
[handle like event data files]
Application and smaller auxiliary files
[won't discuss]

Data handling in lots of boxes



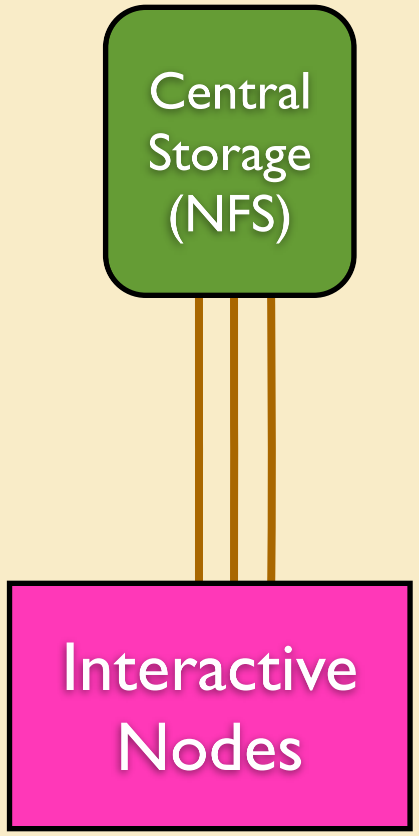

Data input/output with SAM
... to and from worker
Application Code and Auxiliary files

The last mile evolves

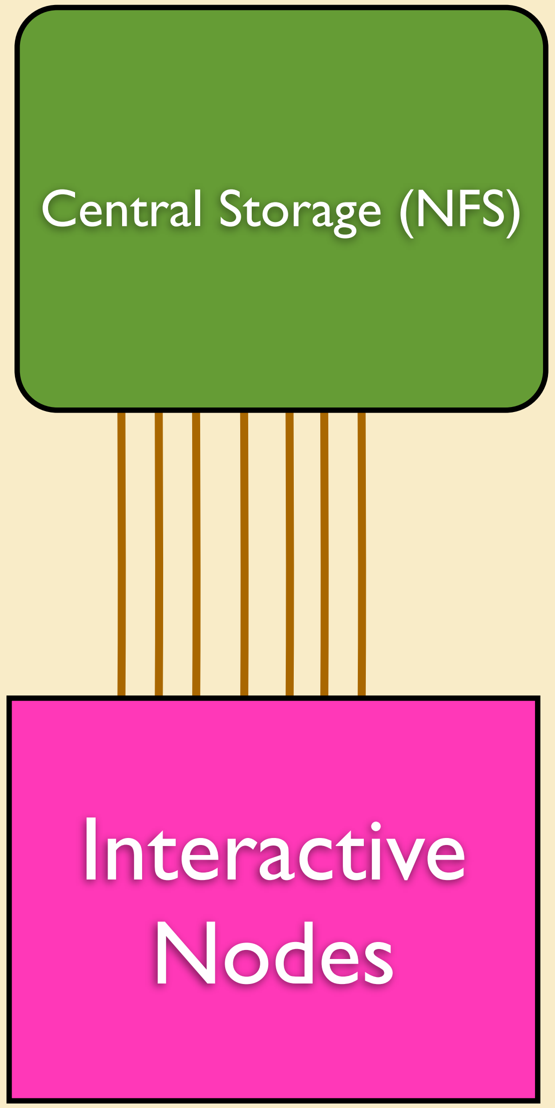


Early days

The last mile evolves

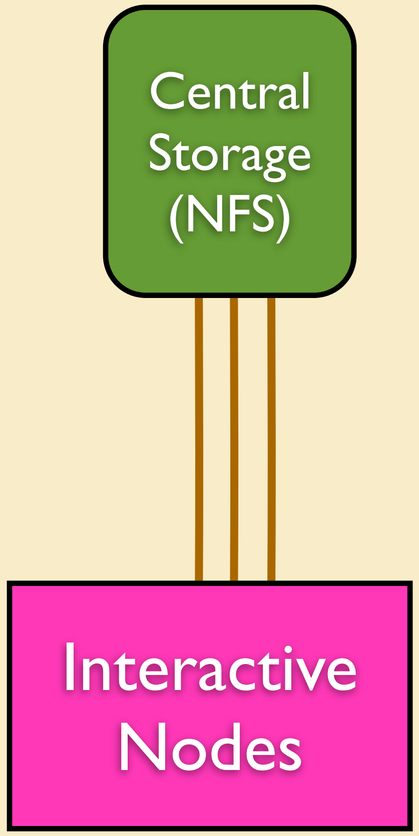


Early days

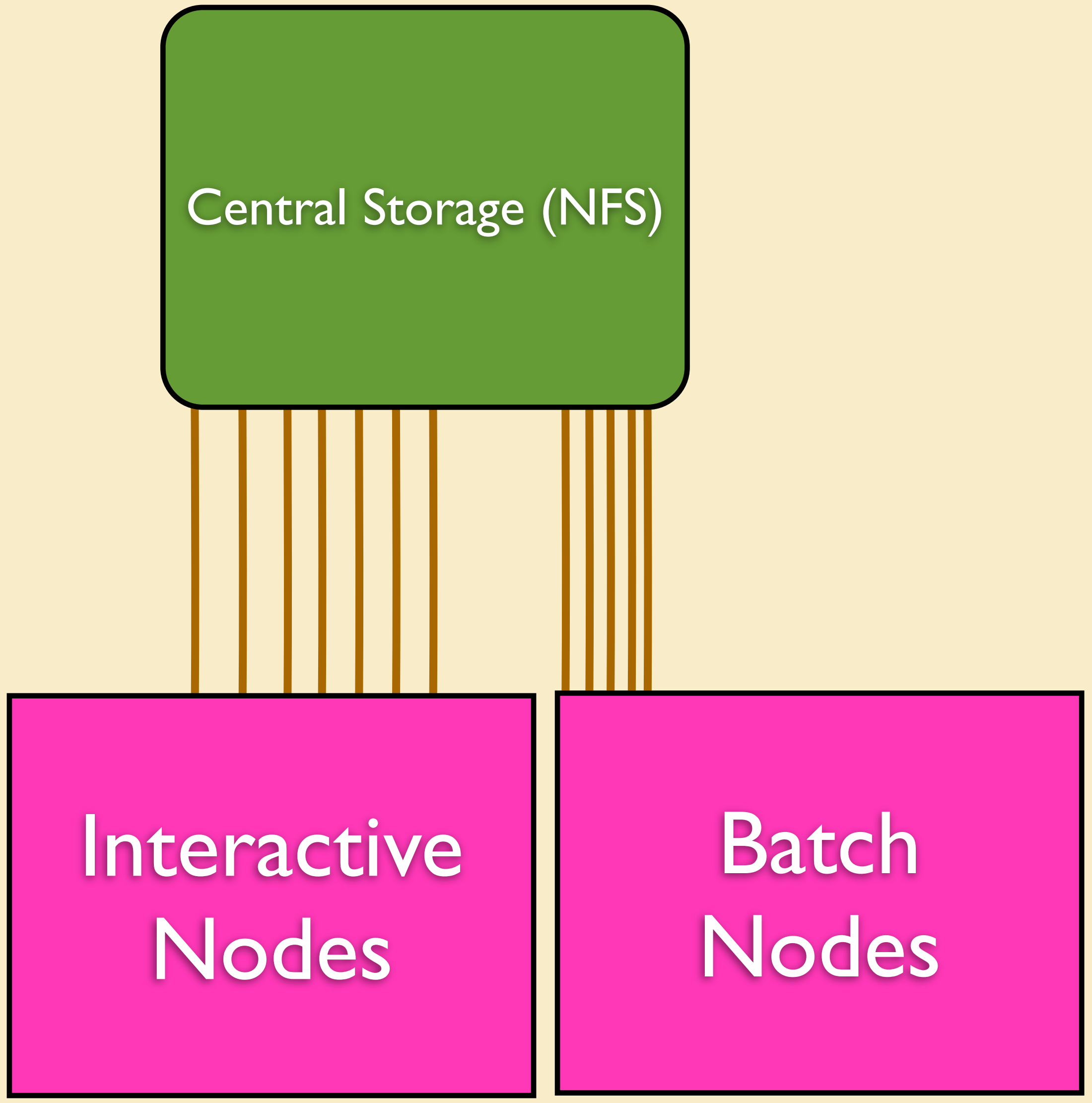


Intermediate

The last mile evolves

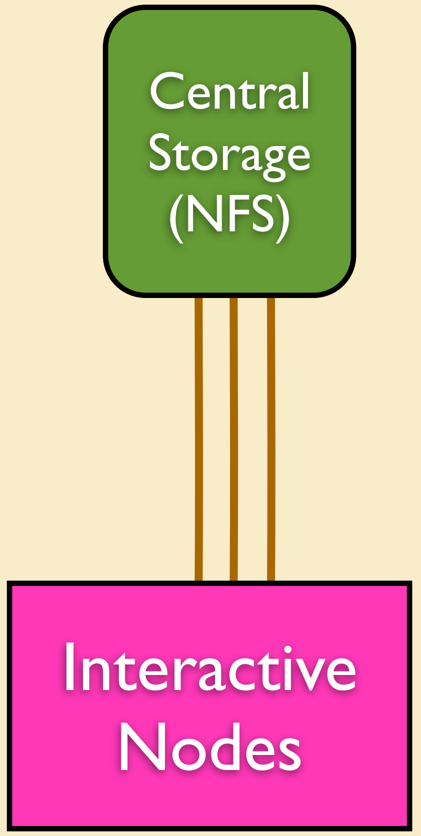


Early days

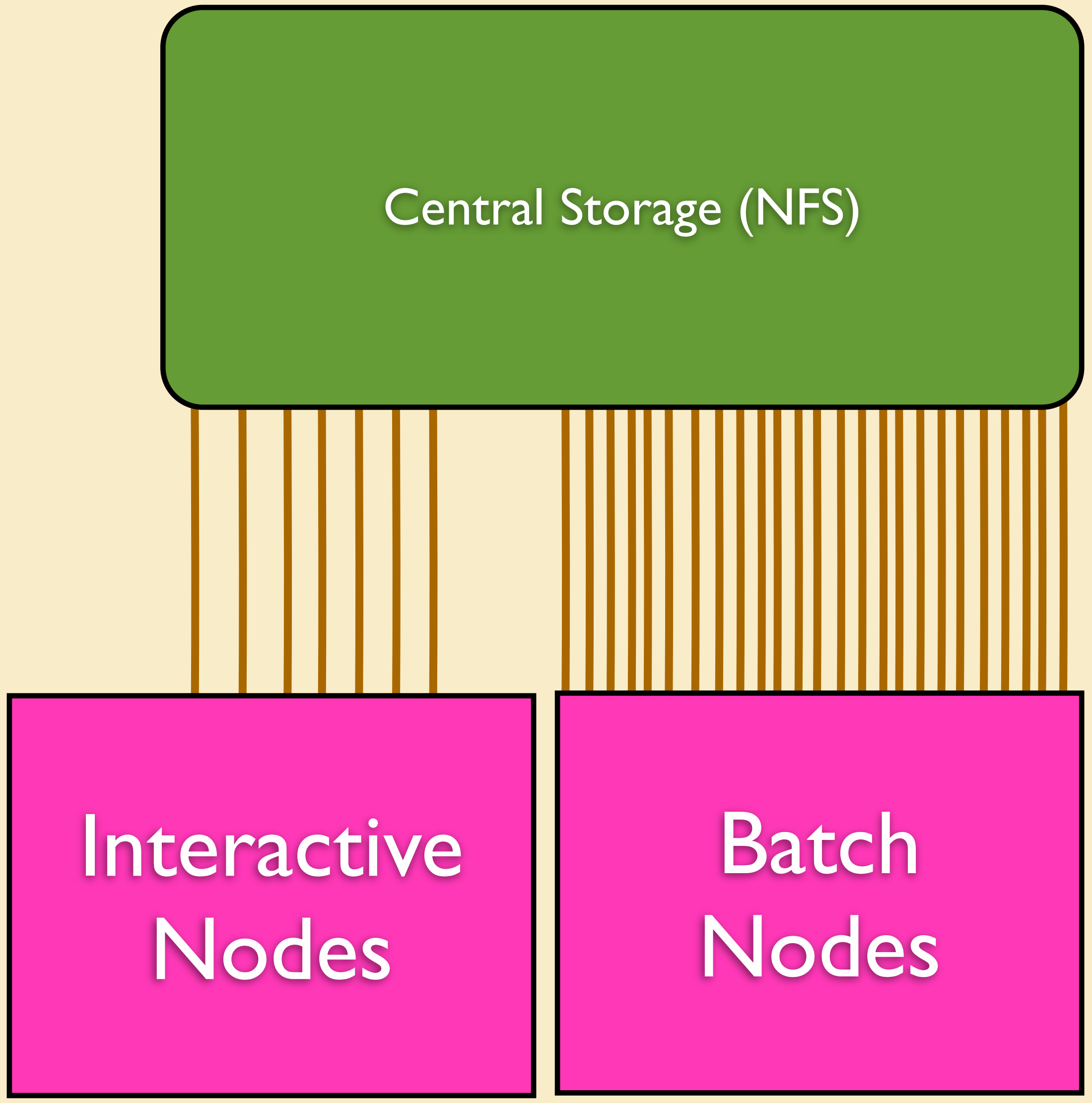


Intermediate; need batch

The last mile evolves

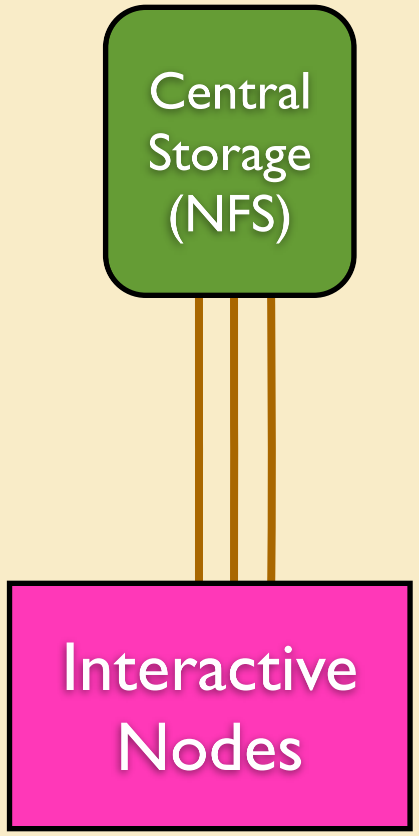


Early days

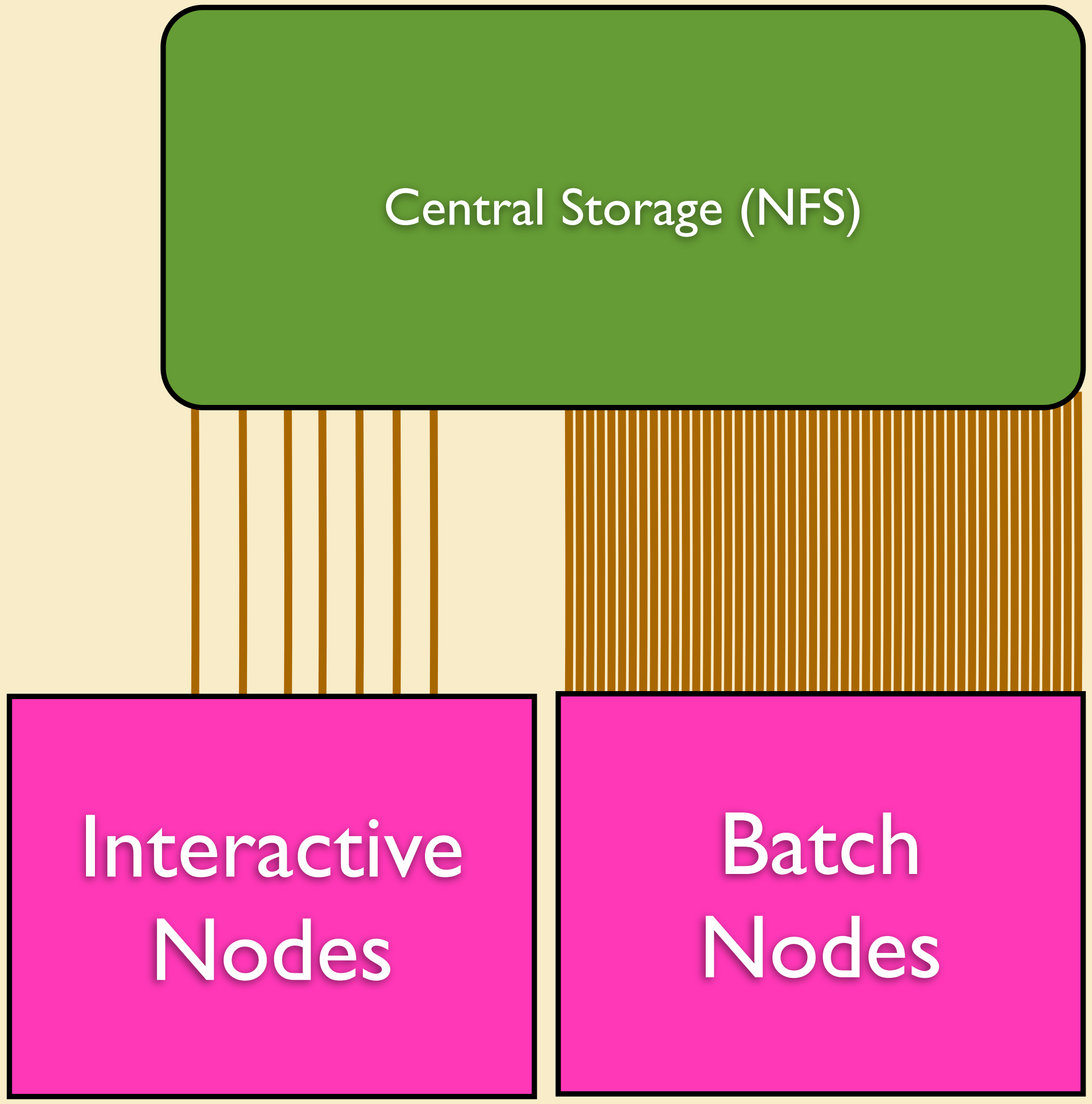


Intermediate; need more batch

The last mile evolves



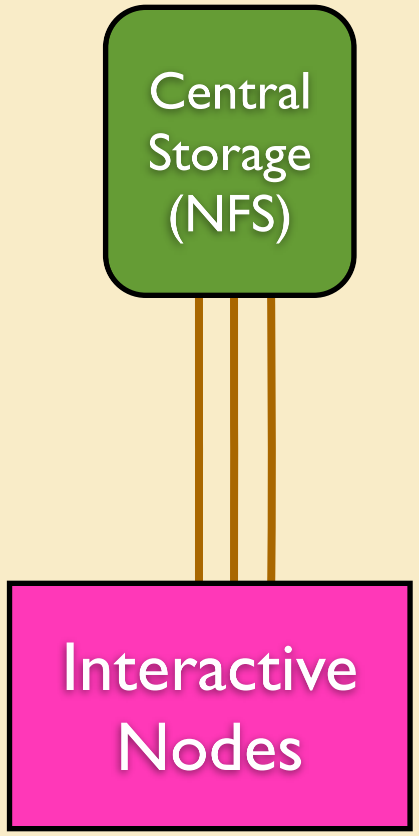
Early days



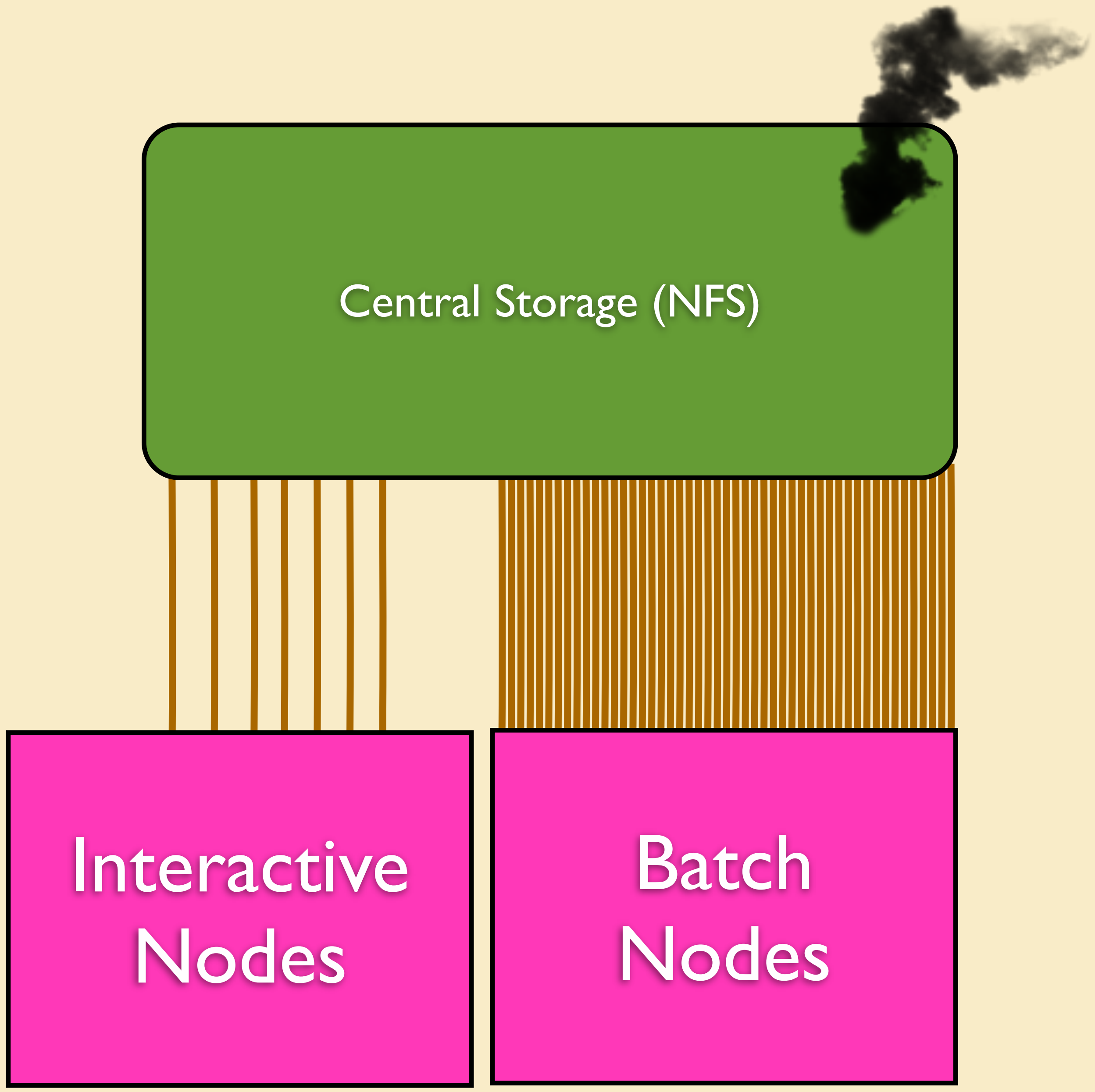
Intermediate; uh oh

scale problems!

The last mile evolves



Early days

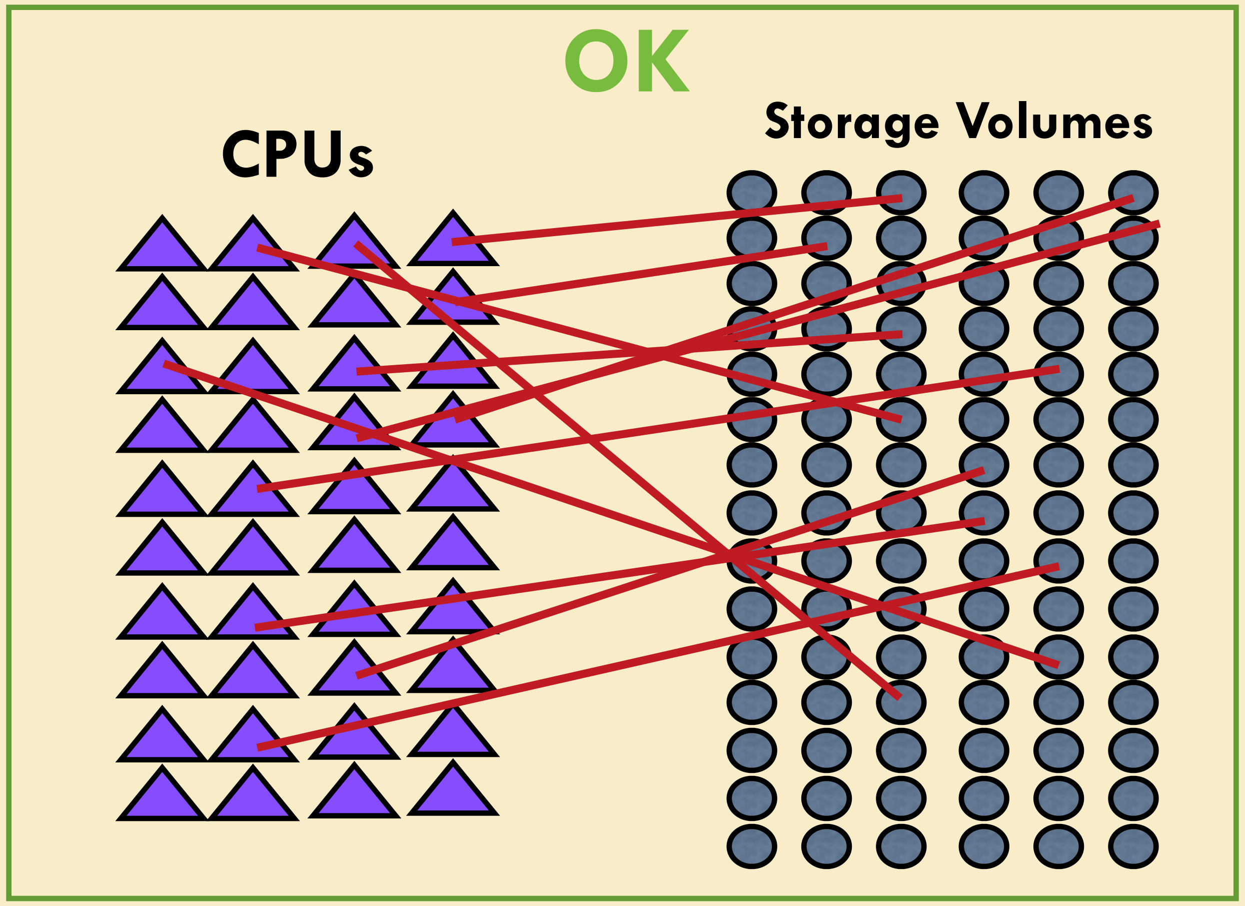


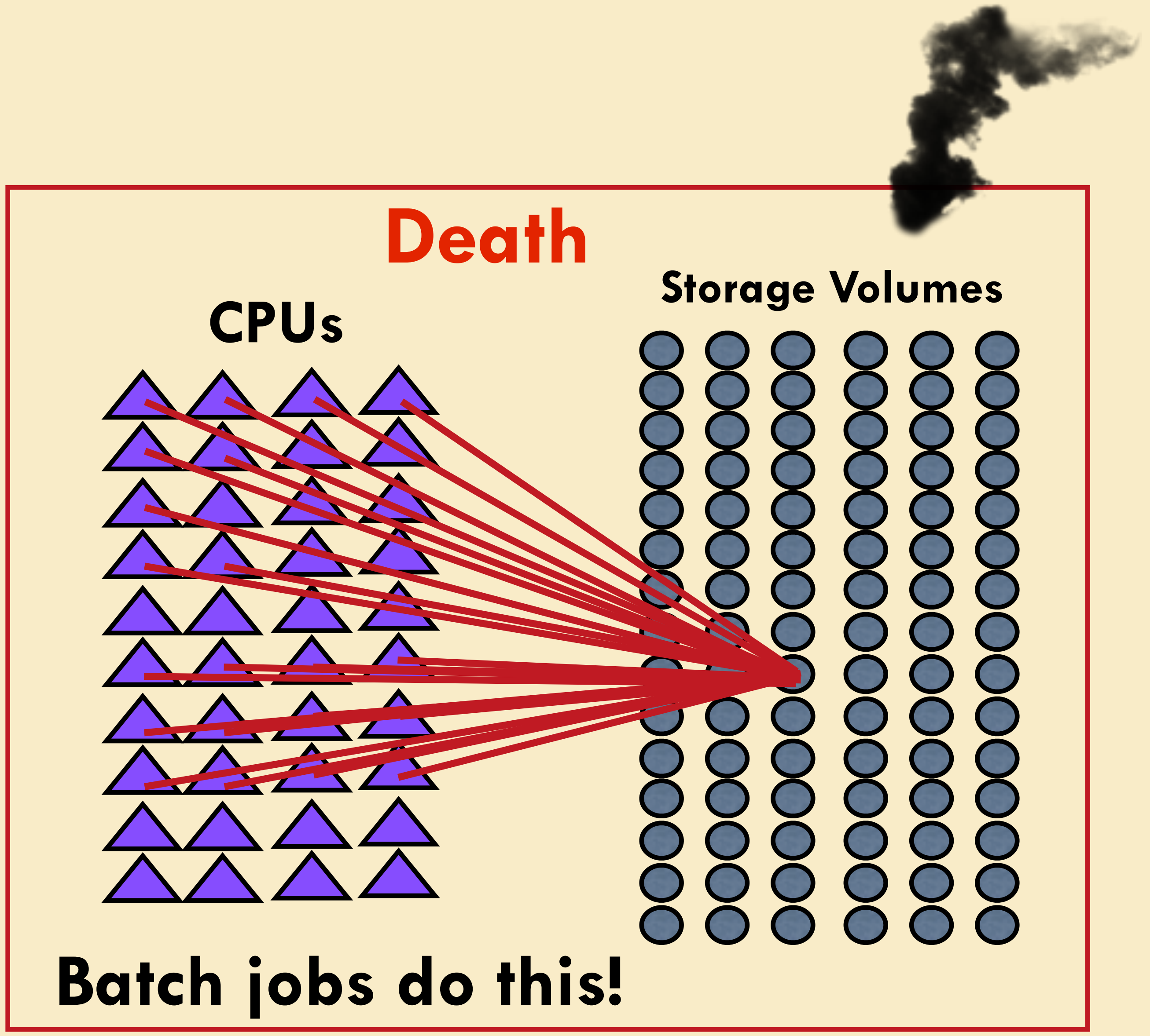
Intermediate; uh oh
scale problems!

Scale problems begin
e.g. Uncontrolled Bluearc

Good news:
When used as designed, it works great

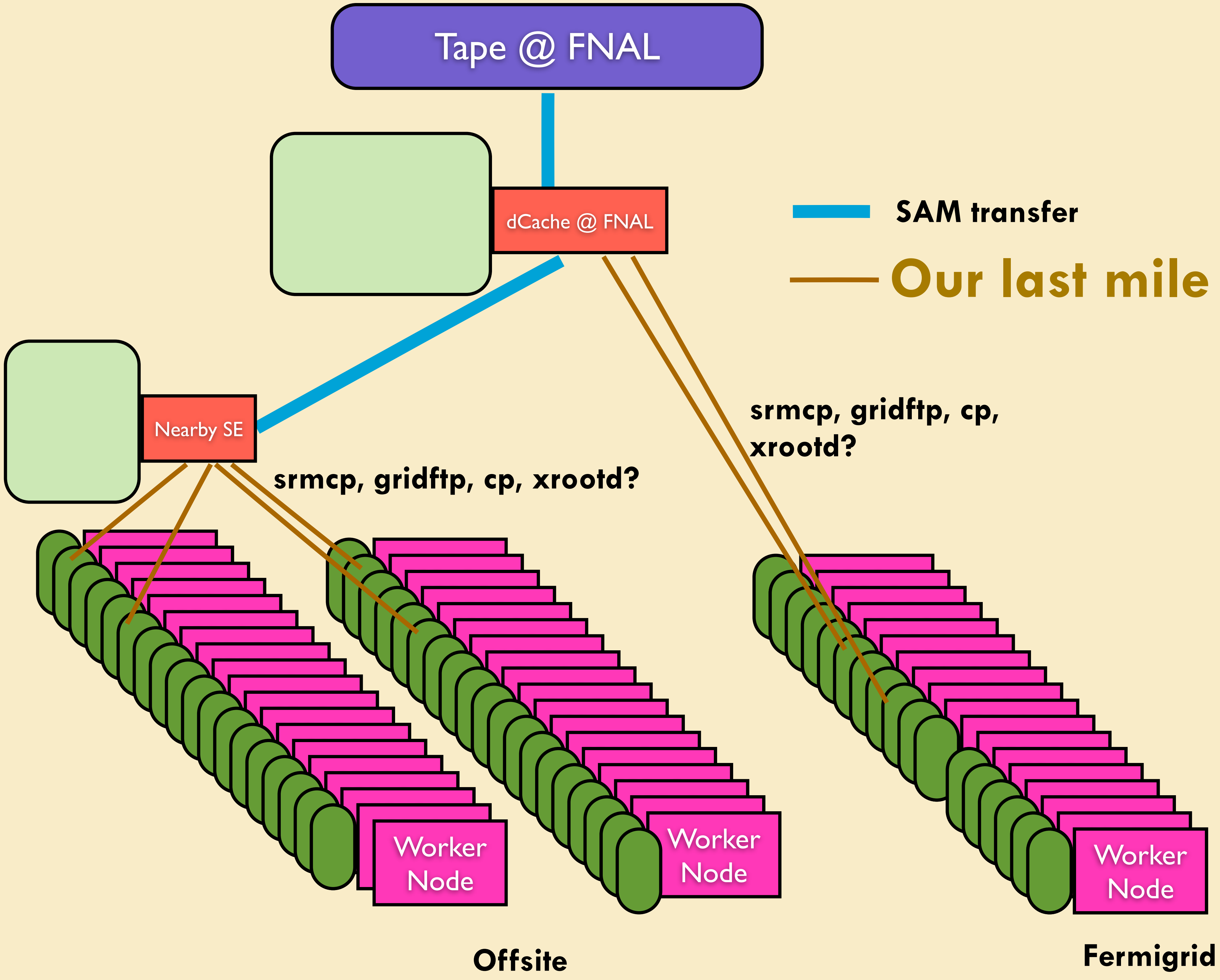
Bad news:
Can't handle concentrated access



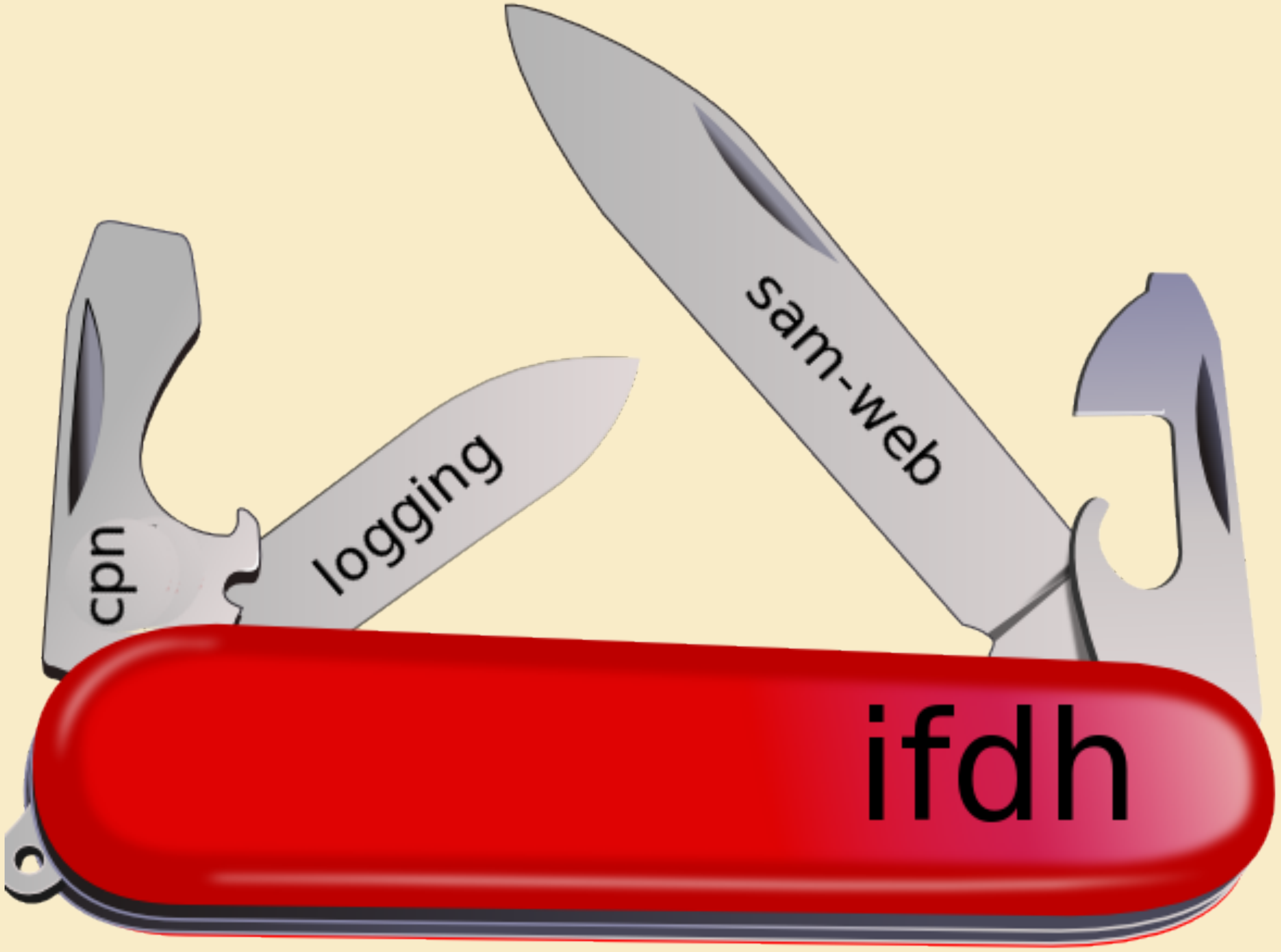


Batch jobs do this!

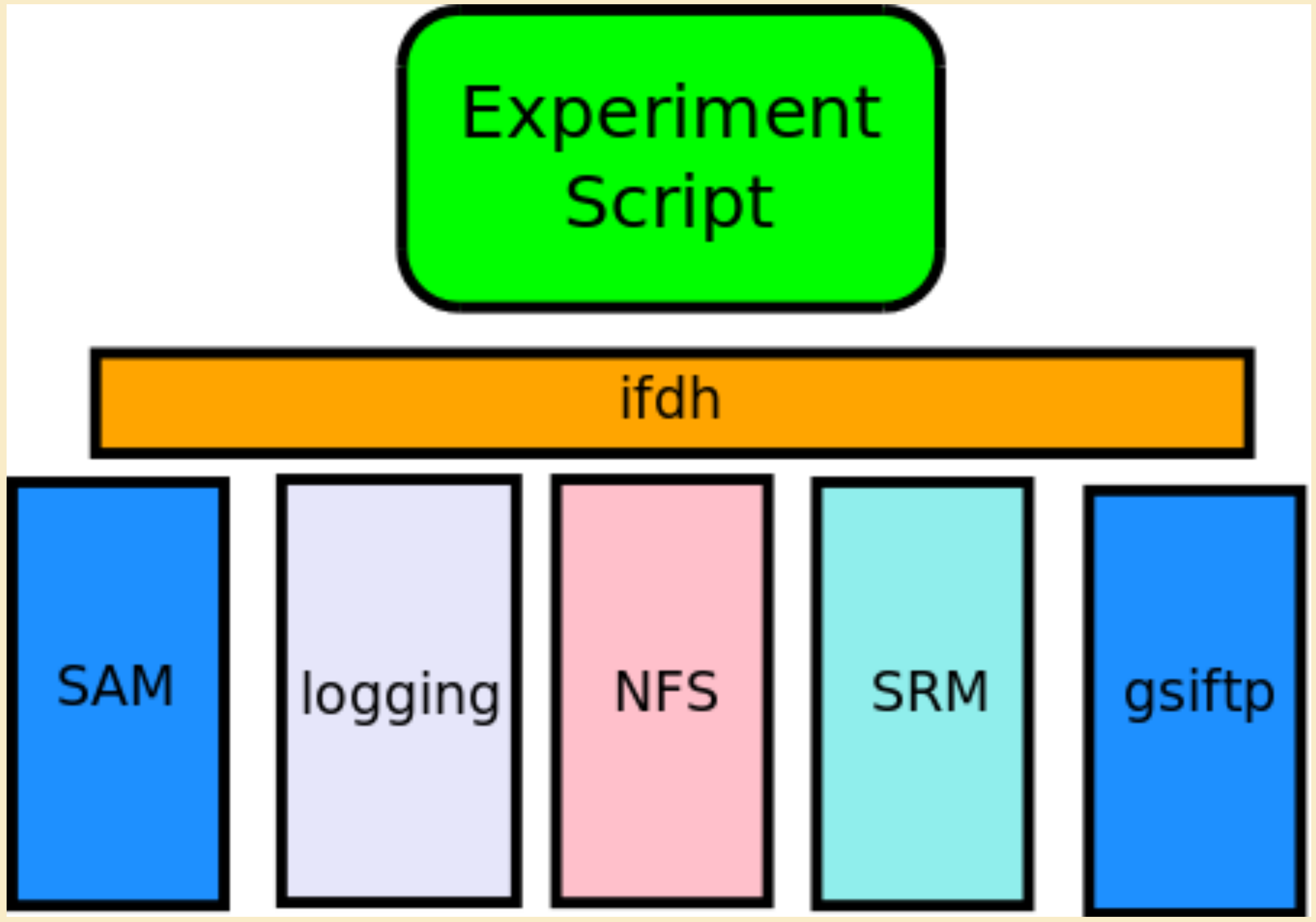
Growing up to real data management



Solution for the last mile - IFDH



IFDH Thin Layer for Data Movement



Automatic protocol discovery or specify with environment var

Protocols supported: cp (throttled), srm, gridftp, xrootd (near future)

Users need not worry about protocol

Users’ job scripts use ifdh for transfers - simple

```
while read sourcefile
do
  ifdh cp $sourcefile localsource
  framework_exe -c config localsource localout
  ifdh cp localout $outarea
done < playlist
```

With a data handling system

```
while uri=`ifdh getNextFile $projectUrl` && [-n “$uri”]
do
  localsource = `ifdh fetchInput $uri`
  framework_exe -c config $localsource localout
  ifdh cp localout $outarea
done
```

With an integrated framework

```
framework_exe -c config --samProjectUrl $projectUrl -o $outarea
ifdh copyBackOutput $myOutArea $outarea
```

Protect the central storage

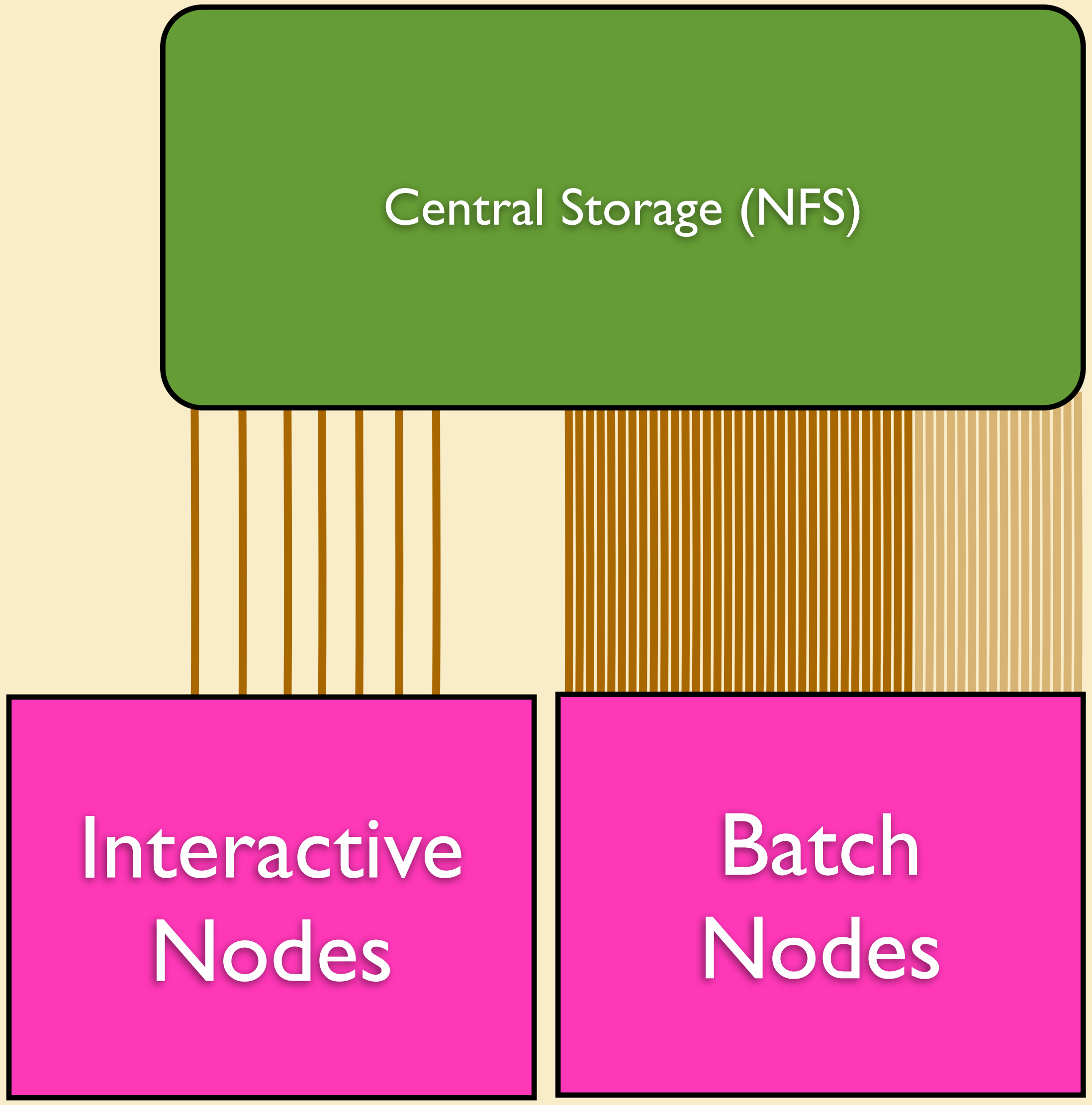
`ifdh cp` to central storage is throttled by a system called "CPN"

Allows n simultaneous transfers per experiment to Bluearc

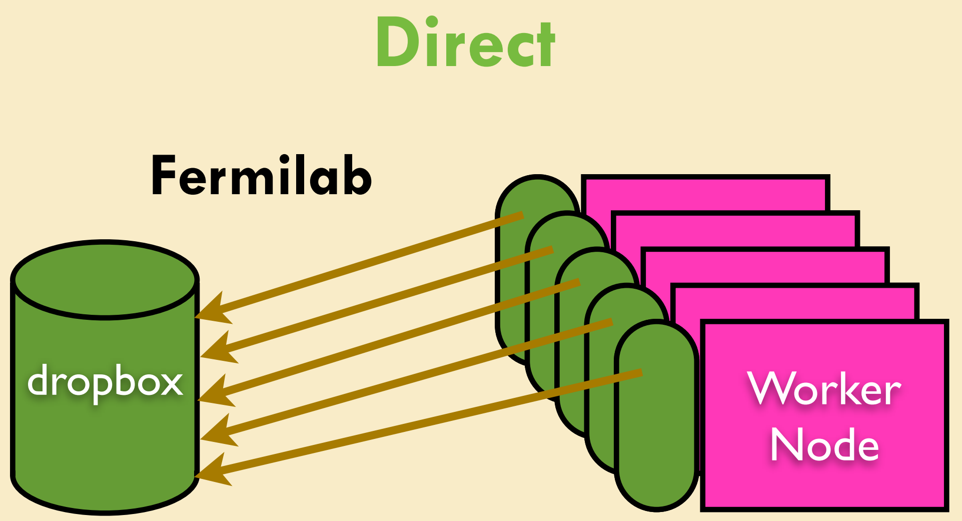
Transfers exceeding are queued

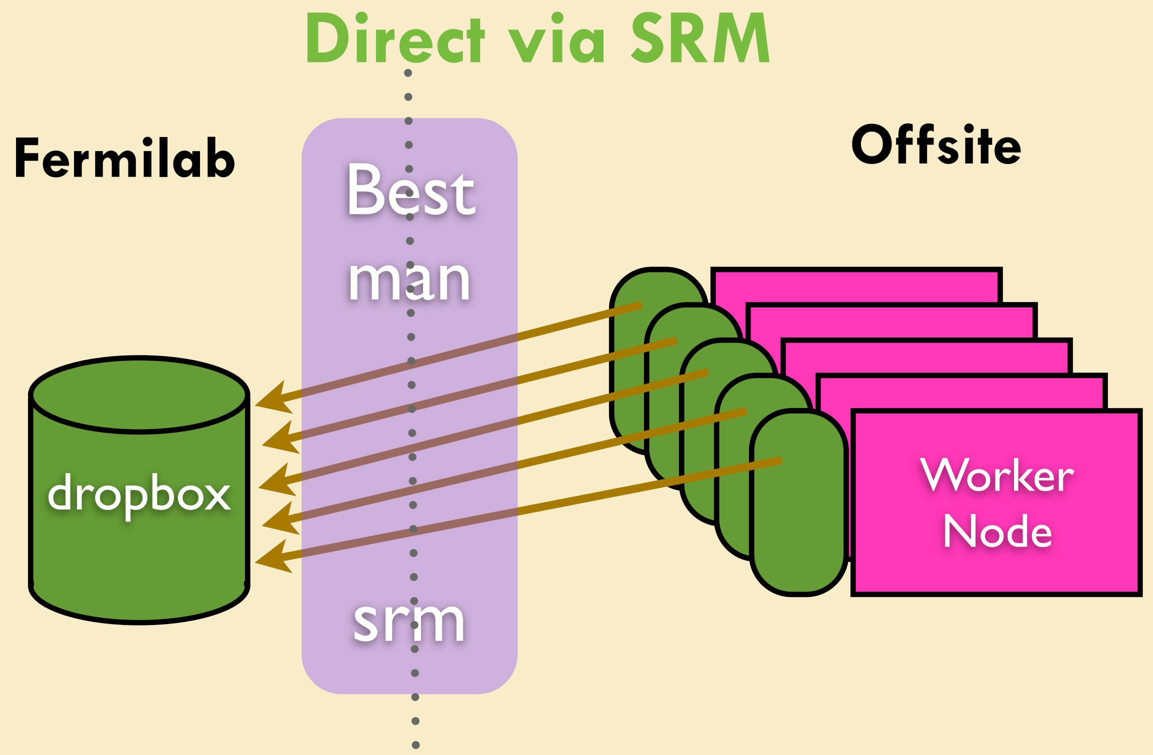
The challenge is to ensure that <u>all</u> users utilize CPN.

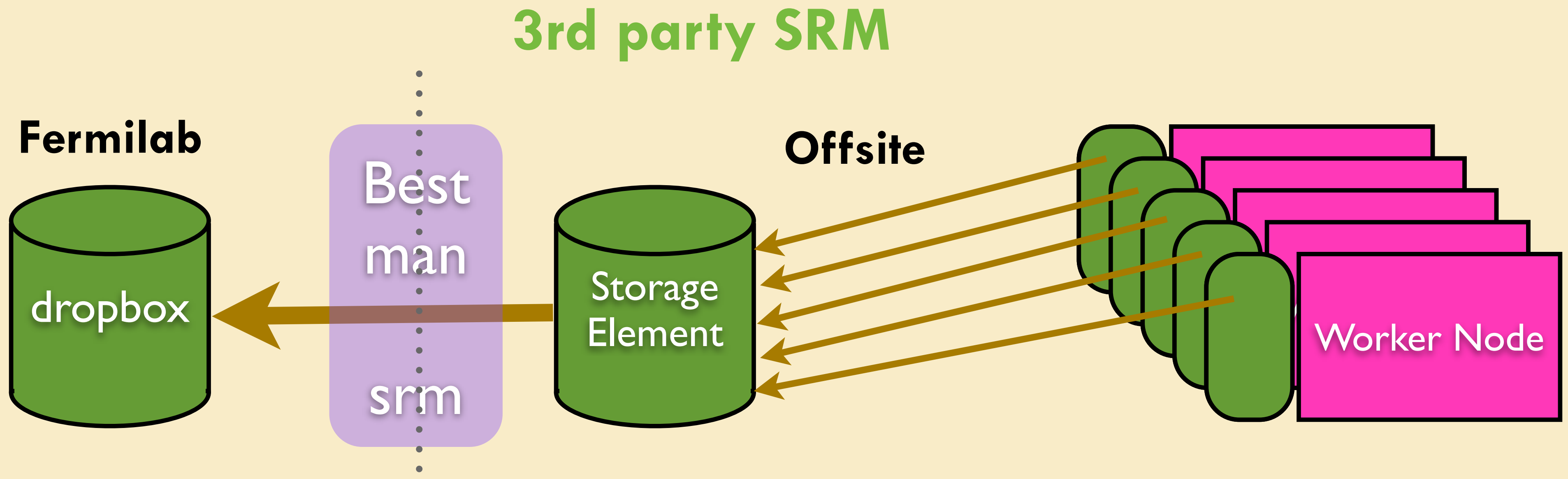
`ifdh cp` makes that easy



Harder part – transferring output back home







Benefits of the thin abstraction

Users are shielded from details of protocol choice

We can change the protocols and decision algorithms without breaking the user's scripts

Shipped to remote sites with CVMFS (but small, can be shipped with job)

Other features:
- o Tools to define and view SAM data-sets
- o Tools to locate files on tape or in cache
- o Logging over UDP to monitoring services
- o Supports many languages: C++, Python, Bash

Examples

C++

```
#include “ifdh.h”
ifdh i();
location = i.locateFile(base_uri, filename);
```

Python

```
import ifdh
i = ifdh.ifdh()
location = i.locateFile(base_uri, filename)
```

Bash shell

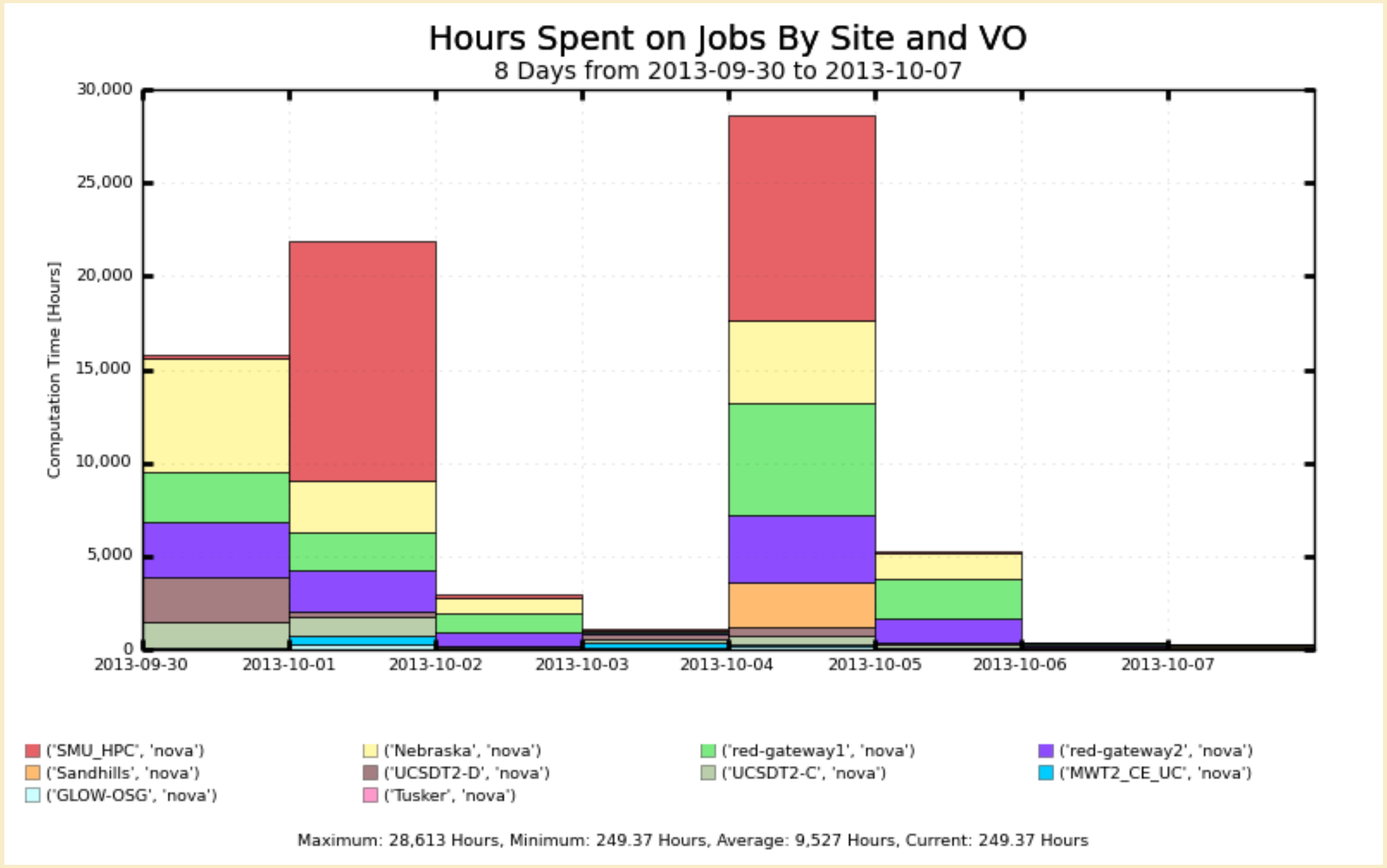
```
location = `ifdh localFile $base_uri $filename`
```

Easy to add more since we use SWIG

Experience

Very successful for local FermiGrid, Bluearc and remote sites

- Greatly reduced Bluearc downtime due to overloads
- NOvA experiment doing MC generation with SAM and ifdh remotely
  Configuration file retrieved by SAM/ifdh; output returned by ifdh



Plans & Summary

Future:
o Explore other protocols as necessary
o Settle on file return feature and discovery

Summary:
IFDH is a swiss-army knife of tools for data movement abstraction and protocol selection for the last mile

Users learn one simple system – shielded from details

Details can be changed without affecting users

IFDH is an integral part of our data management solution